DPI-691M: Programming and Data for Policymakers (January 2019)
Instructor: Dhrumil Mehta <Dhrumil_Mehta@hks.harvard.edu>

Course Overview

Data and code are no longer just for programmers. Policymakers, from members of congress to analysts and executives need to be equipped with the necessary skills to navigate government in the 21st century. This course is built with the understanding that hands-on experience with programming, data, software development and management methods, open source collaboration, and technology innovation will better prepare students to competently navigate technology projects and nuanced issues at the intersection of technology and governance than a theoretical understanding alone.

This J-term module will provide an intensive hands-on curriculum that involves programming exercises in the context of government and politics to build essential core technology and data skills. The hands on exercises will familiarize students with technologies that are a part of the modern programmers toolkit including the command-line, github, the structure of a web application, the linux filesystem, data standards (json, XML, etc), cloud technologies, and databases. Students will work in a team to develop a small software project involving analyzing and visualizing a policy-related dataset of their choice.

This is not a data science course, nor is it a standard programming bootcamp. This course is designed to give students a deeper appreciation for the nuances of software and data through practical training. Though the course is rooted in practical applications, students will be prompted through assigned readings and discussion to think broadly about the implications of implementing the technologies introduced and their ramifications in the context of policy. The course is also designed to prepare students who wish to pursue further learning to overcome the initial barrier to learning programming and more efficiently direct future efforts.

Expectations & Logistics

There will be two sections, A & B. The class schedule is as follows:

Green = Section A Blue = Section B			Th 1/3	Fr 1/4	Sat	Sun
			10am-6pm	10am-6pm	Project Work	
Mo 1/7	Tu 1/8	We 1/9	Th 1/10	Fr 1/11	Sat	Sun
10am-6pm	10am-6pm	10am-6pm	10am-6pm	Project Work		
Mo 1/14	Tu 1/15	We 1/16	Th 1/17	Fr 1/18	Sat	Sun
10am-6pm	10am-6pm	10am-6pm	10am-6pm	Project Work		

Class will be held on alternating days. There is no class scheduled on weekends, however, students will be expected to get together with their groups and work on their projects and take time to digest the course material during that time. The instructor and classroom will be available on Friday for groups to get together, work on their projects, and ask questions as needed. If you're new to programming, it may be a good idea not to over-schedule your weekends. Projects will be due the weekend of January 18.

Exercises will build on one another towards a final project, which will be a small dynamic web application. Students will be graded on the final projects as well as daily assignments both during and after class.

Readings will be assigned to encourage students to think about policy implications of the technologies they are working with and some time will be dedicated to discussing the readings. Students will be expected to read assigned course materials and come to class ready with discussion questions.

No prior programming experience is necessary, but the course will be rooted in programming exercises and students must be prepared for hands-on learning. Students who have no experience with topics covered may consider staying after class or coming in on the weekend for individual instruction or help with debugging code.

Course Philosophy

- Rolling up their sleeves and writing some code will give students a sense of what all the basic parts of software are and how they fit together. This appreciation for code will allow students to better understand and manage technology projects even if they do not intend to work in a role where they are regularly writing code.
- Practical programming and data skills can allow policymakers to be more self-sufficient with regards to basic programming, data collection, and analysis. For students who already have a quantitative background, the course will help fill the gap between having quantitative analysis skills and being able to use them in the real world where data is messy or hard to get.
- Participating in the ecosystem of open source software and understanding the spirit of transparency, civic technology, and agile software development methods will give students a better understanding of where some of the technology-related bottlenecks in government are and how they can be addressed.
- Students will become familiar with modern programming technologies and overcome the initial uphill battle that comes before learning to code. If students chose to pursue further learning, with this basic toolkit in hand, they will be better positioned to identify technologies that will actually be useful for their needs, and learn those technologies in a more efficient and practical way.

What this course is not

- It is not a general programming curriculum (the kind you can find on CodeCademy, etc.)
- It is not an intro to computer science course - we won't be talking about data structures, algorithms, recursion, etc etc… (although some basic concepts such as functions, looping, conditional statements etc will be embedded into the exercises)
- It is not a specific programming language course - rather than teaching a particular language, students will get a glimpse of many different languages as they are found in the wild. They will come away with a toolkit to utilize other resources to learn a language quickly and efficiently if they wish to.

Project

- All of the exercises in the course will build towards the development of a project aimed at using all the skills learned in the course to deploy a small dynamic web application.
- Projects will be done small groups depending on the class size. Each group will be arranged into a "scrum" team.
- Each individual's contribution to the project can also be tracked via github and trello to ensure everyone's participation. Both platforms will be used for project planning and execution but will also serve as a communication tool within the team and with the instructor.
- Projects will be due on **Jan. 20, 2019**. Students will be expected to use the days that class is not in session to arrange time to work on their projects. The instructor will be available to answer questions and for technical help.
- The project will be graded in two parts:
    1. The product: successfully deploying a web application that meets "user" needs
    2. The team's processes as measured by communications on github and trello.

Other Artifacts:

- A small dynamic web application connected to a database
- A web-server on which the application will be deployed
- A static web application / personal website
- A script to scrape data from the web
- A github account which will serve as a portfolio of their technical work completed in the course as well as any future programming students chose to pursue

What we will cover

Day 1: Basics

- Fundamentals of the Unix and the Command Line
- Github and open source collaboration
- Deploy a static web application on github

Day 2: The Web

- Fundamentals of the Web (HTTP, HTML/CSS/Javascript, architecture of a website)
- Collecting data via APIs and web-scraping
- Understanding how data oriented websites are structured

Day 3: Tools for Self Reliance

- Data cleaning and manipulation with regular expressions, command line data tools, and python.
- Skills that students will need to go from data found in the wild to data that is clean enough to analyze it in their tool of choice (excel, stata, R, etc…)

Day 4: Visualizing Data for the Web

- Scaffolding a Model-View-Controller web application
- Learning the fundamentals of javascript-based web visualizations, and existing code to to visualize a dataset of your choice

Day 5: Building a Dynamic Web Application

- Databases - SQL (and briefly noSQL and alternative database types)
- Deploying a dynamic web application on Amazon EC2
- Exploring an unfamiliar codebase

Example Day

Pre-work:

- Make a github account
- Reading (No more than 2 hours)
    - Selected excerpts from canonical texts on open source philosophy
    - Short blog-posts by practitioners that speak to the promises and barriers of open source technologies in government
    - Read an open source license

Classtime

- 30 mins - Daily scrum meeting: quick updates on project progress
- 30 mins - Introductions and setup
- 1 hr - Discussion of readings
- 1 hr - Fundamentals of the Unix Command Line - building command line muscle memory through short exercises.
- 2 hrs - Github
    - Set up ssh keys, and learn about security
    - Issue your first pull request, participate in exercises where you learn how to push code and collaborate on github (including forking code, working in branches, and other github functions)
    - Serve a basic personal website with github pages, understand github as a personal portfolio, explore code on github for subjects of interest to you
    - Set up the github repository for the course project
- 1 hour - Debate and Q&A
    - a guest speaker who works with open source technology in government will host a debate with students in which students argue the merits or limitations of open source software. This will be followed by a Q&A to discuss the current state of open source technologies and modern development tools in government and challenges to their implementation.

- After Class - Instructor will be available for an hour or two after class to help debug any problems or provide one-on-one instruction. Students may use this time and space to work on their projects.

Assignments

- Assignment that involves practicing the skills learned in class in a different context (fork a repo, make a branch, make some changes that involve practicing command line commands, merge branch in, issue pull request back)

Grading

- Final Project 40%
    - Part 1: The Product (30%)
    - Part 2: The Process (10%)
    - For both parts, you will be graded as a team, however individual participation will be taken into account as well.
- Daily Assignments 20% - in and after class assignments as well as pre-class reading
- Class Participation 40%
    - In class assignments, participation in class, participation on github, trello, and slack as well.

Academic Honesty

Students are encouraged to work collaboratively and in groups. Each student's individual work will be tracked through their commits to GitHub, participation in discussions, submissions of assignments, and through other means. Academic dishonesty will not be tolerated.

Note

Please direct any questions to Dhrumil Mehta <Dhrumil_Mehta@hks.harvard.edu>